Deep Linking Metrics and Traces

with OpenTelemetry, OpenMetrics, Prometheus and M3

San Diego, 2019-11-21
Rob Skillington

Who

Rob Skillington

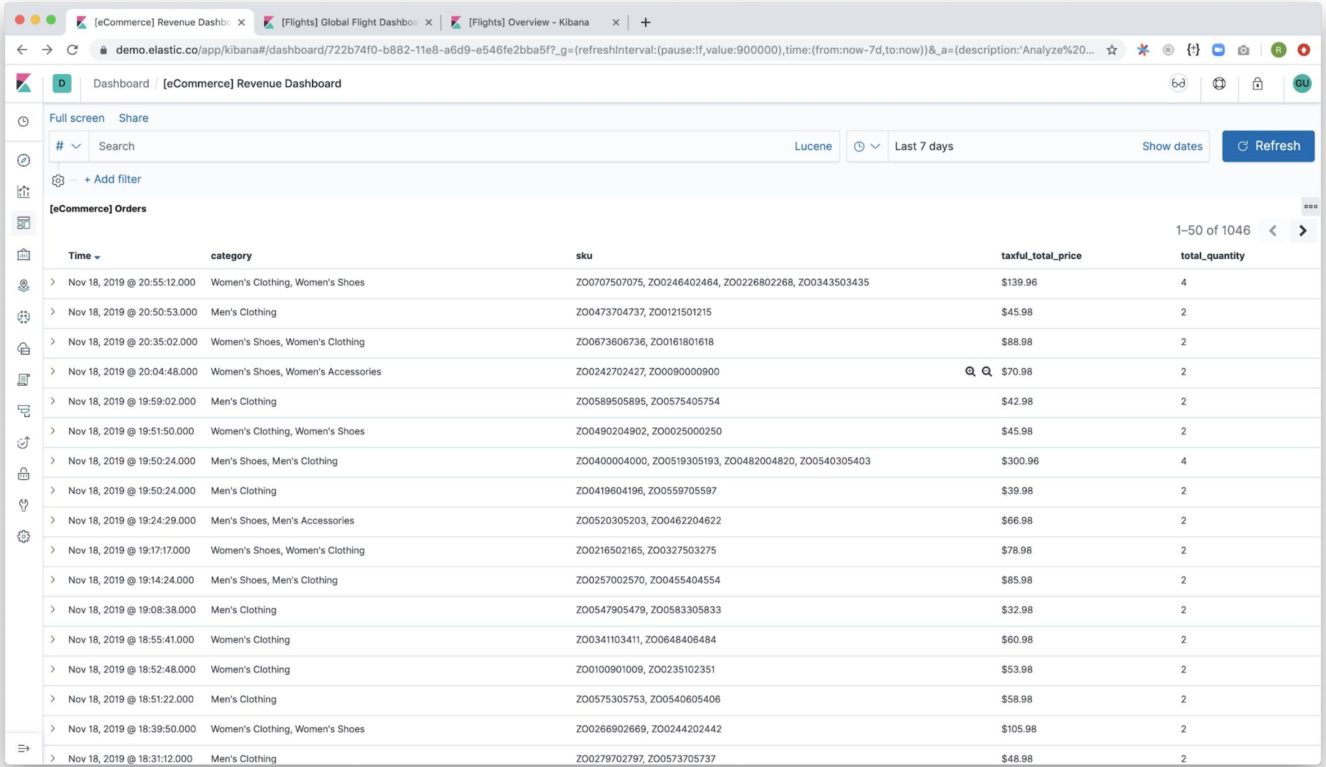
CTO at Chronosphere
Previously M3 and M3DB technical lead at Uber
OpenMetrics Contributor

Let’s talk about

1. State of Monitoring: Logs, Metrics, Traces
2. Combining Logs, Metrics and Traces Today
3. Deep Linking Metrics and Traces

1 State of Monitoring: Logs, Metrics, Traces

Logs



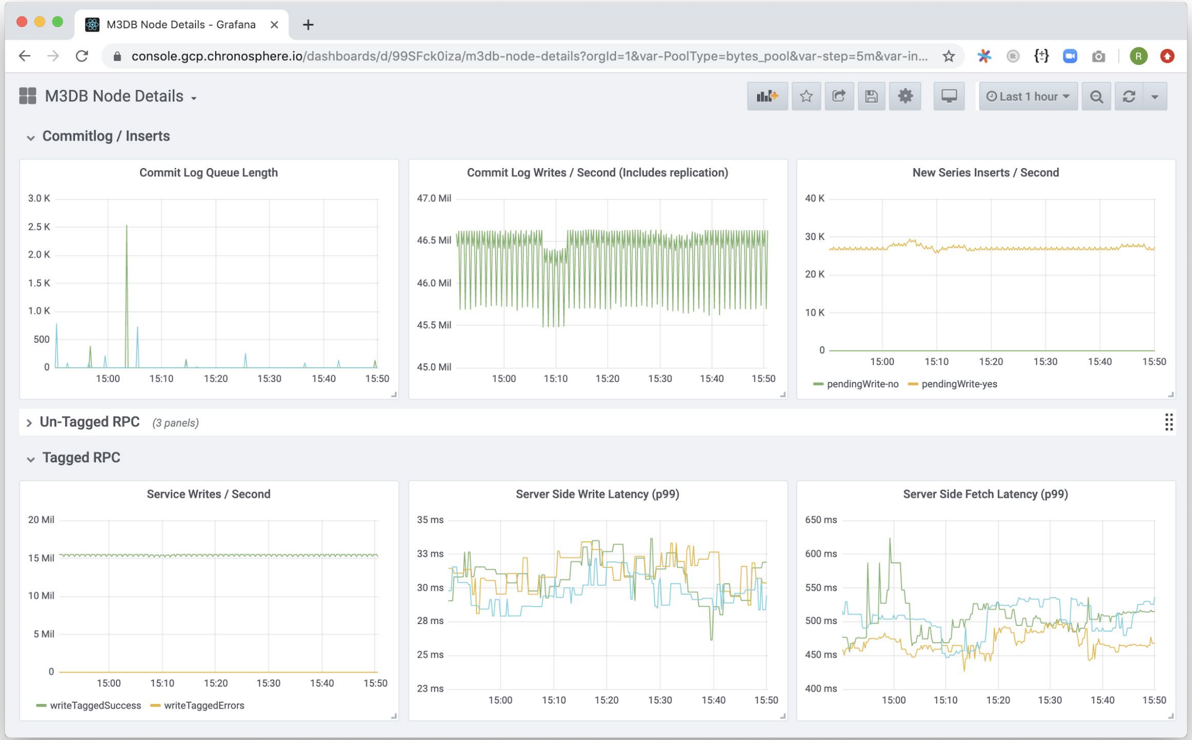
[eCommerce] Revenue Dashbo
[Flights] Global Flight Dashboa
[Flights] Overview - Kibana
demo.elastic.co/app/kibana#/dashboard/722b74f0-b882-11e8-a6d9-e546fe2bba5f?_g=(refreshInterval:(pause:!f,value:900000),time:(from:now-7d,to:now))&_a=(description:'Analyze%20...
Dashboard / [eCommerce] Revenue Dashboard
Full screen
Share
Search
Lucene
Last 7 days
Show dates
Refresh
+ Add filter

[eCommerce] Orders

1–50 of 1046

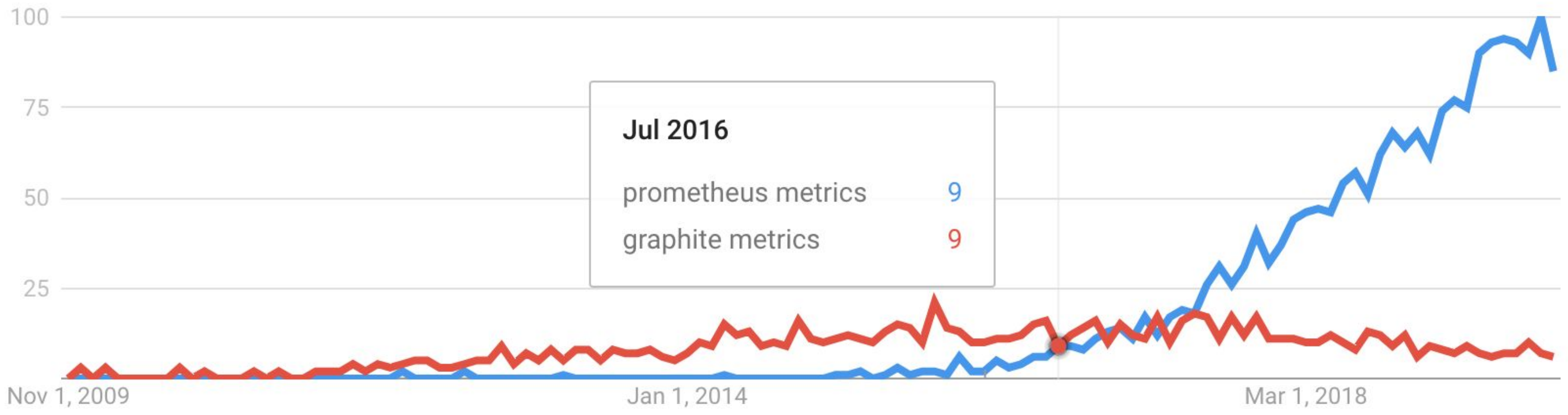
Time	category	sku	taxful_total_price	total_quantity
Nov 18, 2019 @ 20:55:12.000	Women's Clothing, Women's Shoes	ZO0707507075, ZO0246402464, ZO0226802268, ZO0343503435	$139.96	4
Nov 18, 2019 @ 20:50:53.000	Men's Clothing	ZO0473704737, ZO0121501215	$45.98	2
Nov 18, 2019 @ 20:35:02.000	Women's Shoes, Women's Clothing	ZO0673606736, ZO0161801618	$88.98	2
Nov 18, 2019 @ 20:04:48.000	Women's Shoes, Women's Accessories	ZO0242702427, ZO0090000900	$70.98	2
Nov 18, 2019 @ 19:59:02.000	Men's Clothing	ZO0589505895, ZO0575405754	$42.98	2
Nov 18, 2019 @ 19:51:50.000	Women's Clothing, Women's Shoes	ZO0490204902, ZO0025000250	$45.98	2
Nov 18, 2019 @ 19:50:24.000	Men's Shoes, Men's Clothing	ZO0400004000, ZO0519305193, ZO0482004820, ZO0540305403	$300.96	4
Nov 18, 2019 @ 19:50:24.000	Men's Clothing	ZO0419604196, ZO0559705597	$39.98	2
Nov 18, 2019 @ 19:24:29.000	Men's Shoes, Men's Accessories	ZO0520305203, ZO0462204622	$66.98	2
Nov 18, 2019 @ 19:17:17.000	Women's Shoes, Women's Clothing	ZO0216502165, ZO0327503275	$78.98	2
Nov 18, 2019 @ 19:14:24.000	Men's Shoes, Men's Clothing	ZO0257002570, ZO0455404554	$85.98	2
Nov 18, 2019 @ 19:08:38.000	Men's Clothing	ZO0547905479, ZO0583305833	$32.98	2
Nov 18, 2019 @ 18:55:41.000	Women's Clothing	ZO0341103411, ZO0648406484	$60.98	2
Nov 18, 2019 @ 18:52:48.000	Women's Clothing	ZO0100901009, ZO0235102351	$53.98	2
Nov 18, 2019 @ 18:51:22.000	Men's Clothing	ZO0575305753, ZO0540605406	$58.98	2
Nov 18, 2019 @ 18:39:50.000	Women's Clothing, Women's Shoes	ZO0266902669, ZO0244202442	$105.98	2
Nov 18, 2019 @ 18:31:12.000	Men's Clothing	ZO0279702797, ZO0573705737	$48.98	2

Metrics


M3DB Node Details - Grafana
console.gcp.chronosphere.io/dashboards/d/99SFck0iza/m3db-node-details?orgId=1&var-PoolType=bytes_pool&var-step=5m&var-in...
M3DB Node Details
Last 1 hour
Commitlog / Inserts
Commit Log Queue Length
Commit Log Writes / Second (Includes replication)
New Series Inserts / Second
pendingWrite-no
pendingWrite-yes
Un-Tagged RPC (3 panels)
Tagged RPC
Service Writes / Second
writeTaggedSuccess
writeTaggedErrors
Server Side Write Latency (p99)
Server Side Fetch Latency (p99)

Metrics

Google trends of popular metrics formats

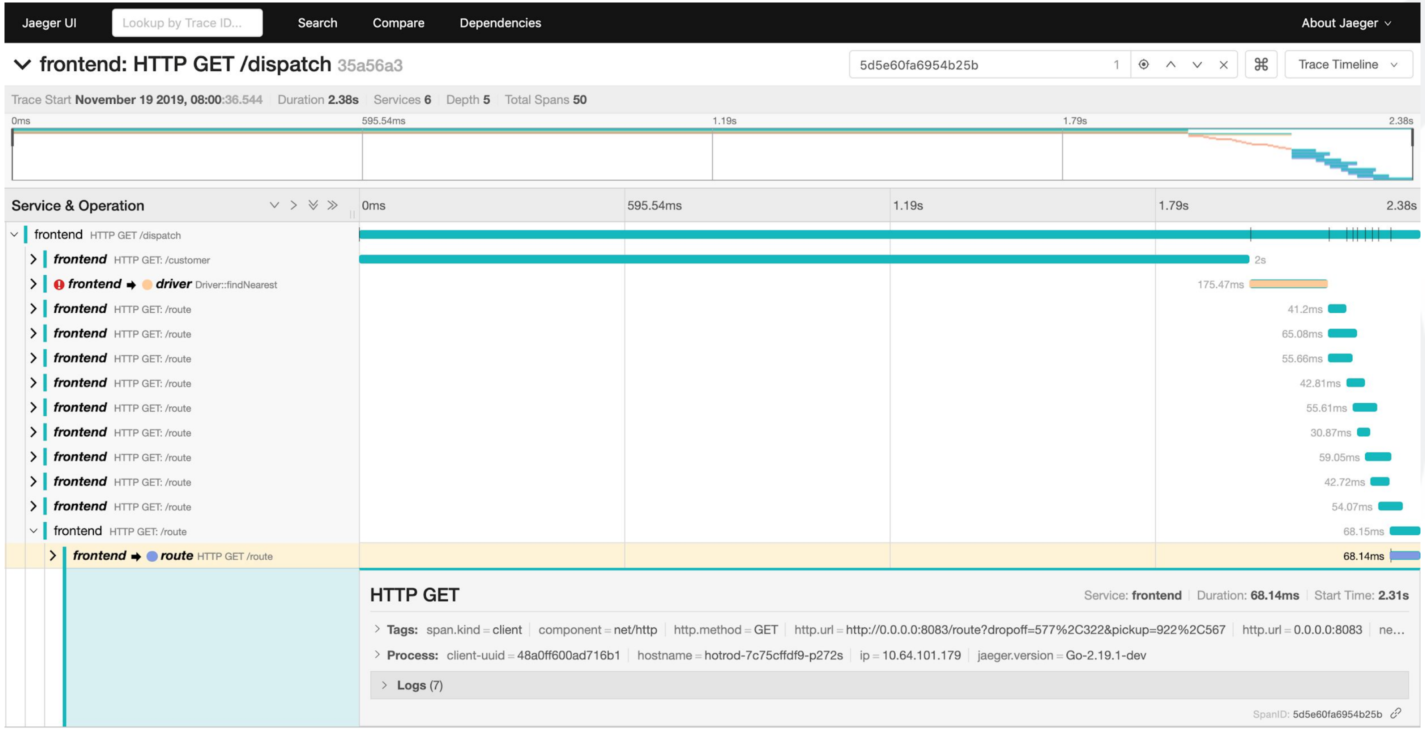


Metrics

Numbers shared at PromCon 2019

2016	2017	2018	2019
~2,800	~16,000	~54,000	~242,000

Tracing



frontend: HTTP GET /dispatch 35a56a3

5d5e60fa6954b25b 1

Trace Start **November 19 2019, 08:00**:36.544 Duration **2.38s** Services **6** Depth **5** Total Spans **50**

0ms 595.54ms 1.19s 1.79s 2.38s

Service & Operation	0ms 595.54ms 1.19s 1.79s 2.38s
frontend HTTP GET /dispatch	
frontend HTTP GET: /customer	2s
frontend ➡ *driver* Driver::findNearest	175.47ms
frontend HTTP GET: /route	41.2ms
frontend HTTP GET: /route	65.08ms
frontend HTTP GET: /route	55.66ms
frontend HTTP GET: /route	42.81ms
frontend HTTP GET: /route	55.61ms
frontend HTTP GET: /route	30.87ms
frontend HTTP GET: /route	59.05ms
frontend HTTP GET: /route	42.72ms
frontend HTTP GET: /route	54.07ms
frontend HTTP GET: /route	68.15ms
frontend ➡ *route* HTTP GET /route	68.14ms

HTTP GET Service: **frontend** Duration: **68.14ms** Start Time: **2.31s**

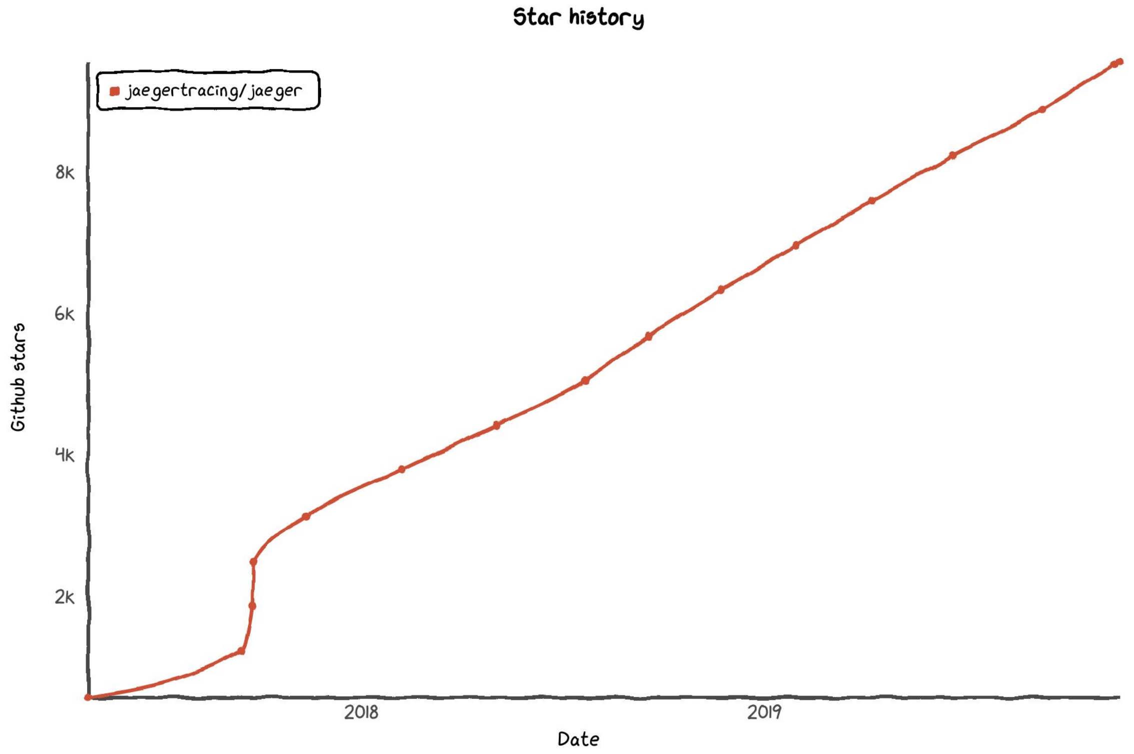
Tags: span.kind = client component = net/http http.method = GET http.url = http://0.0.0.0:8083/route?dropoff=577%2C322&pickup=922%2C567 http.url = 0.0.0.0:8083 ne...

Process: client-uuid = 48a0ff600ad716b1 hostname = hotrod-7c75cffdf9-p272s ip = 10.64.101.179 jaeger.version = Go-2.19.1-dev

Logs (7)

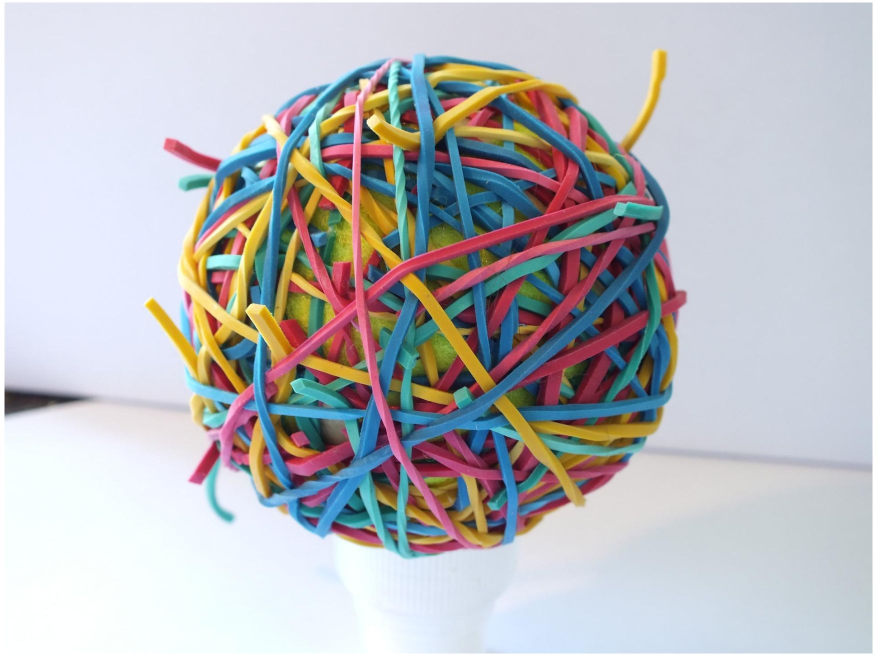
SpanID: 5d5e60fa6954b25b

Tracing

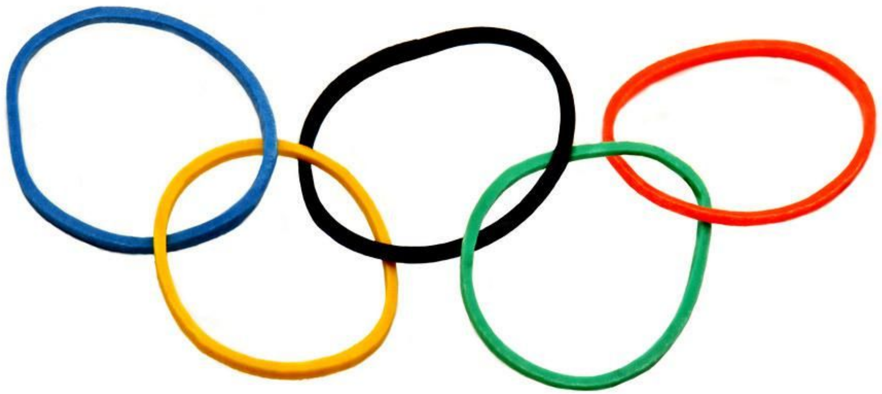


Star history
jaegertracing/jaeger
8k
6k
4k
2k
Github stars
2018
2019
Date

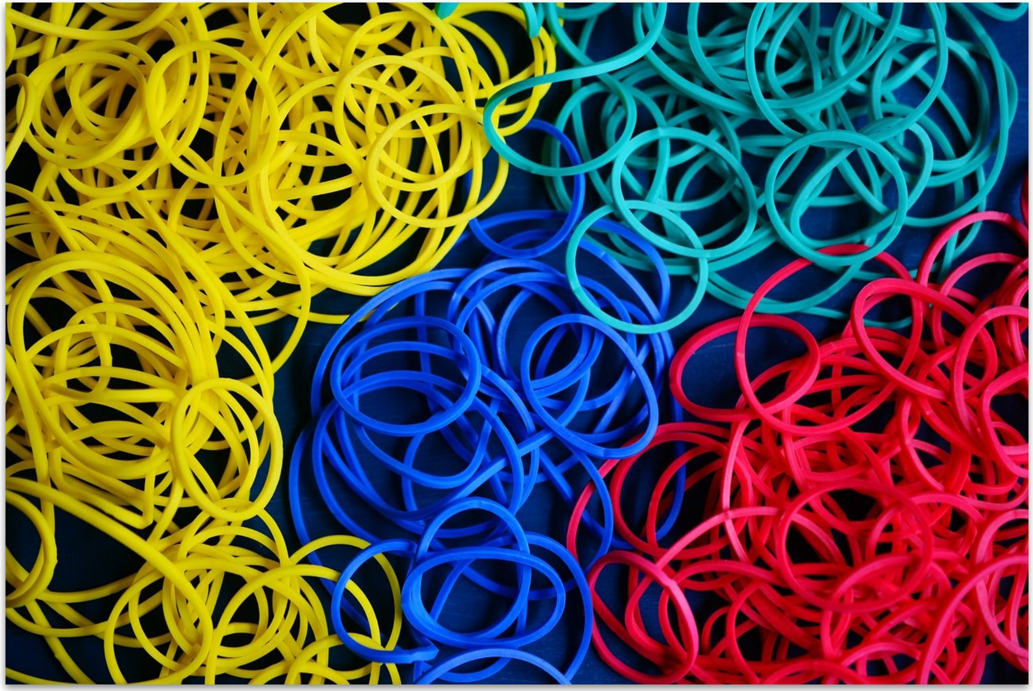
A perspective on... Logs



A perspective on… Traces



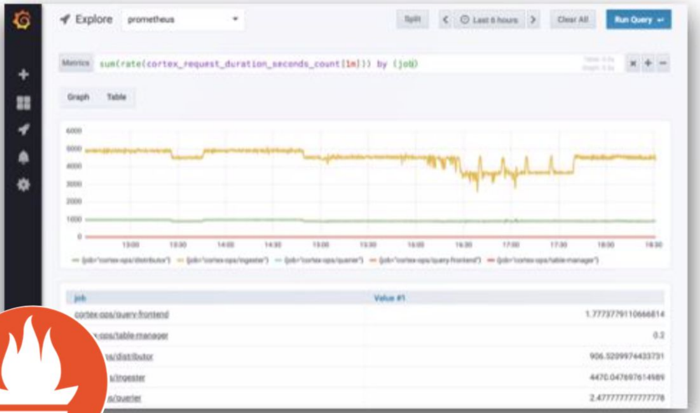
A perspective on... Metrics



2 Combining Logs, Metrics, and Traces Today

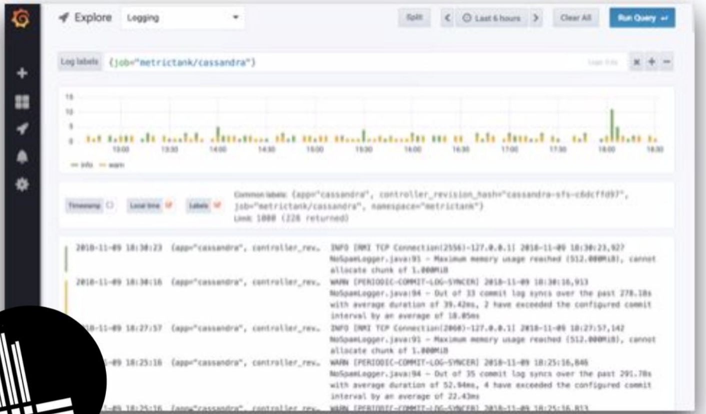
Current integrations

Increasingly, more Observability platforms providing two or more signals (logs, metrics, traces)



```
sum(rate(cortex_request_duration_seconds_count[1m])) by (job)
```

job	Value #1
cortex-ops/query-frontend	1.7773779110666814
x-ops/table-manager	0.2
s/distributor	906.5209974433731
s/ingester	4470.047697614989
s/querier	2.477777777777778



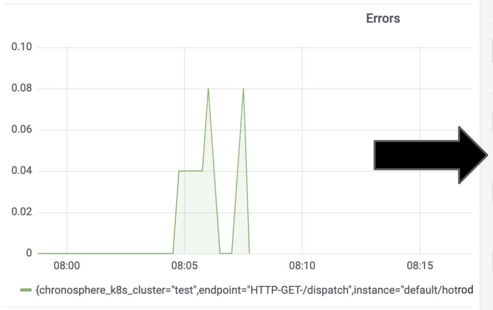
```
{job="metrictank/cassandra"}
```

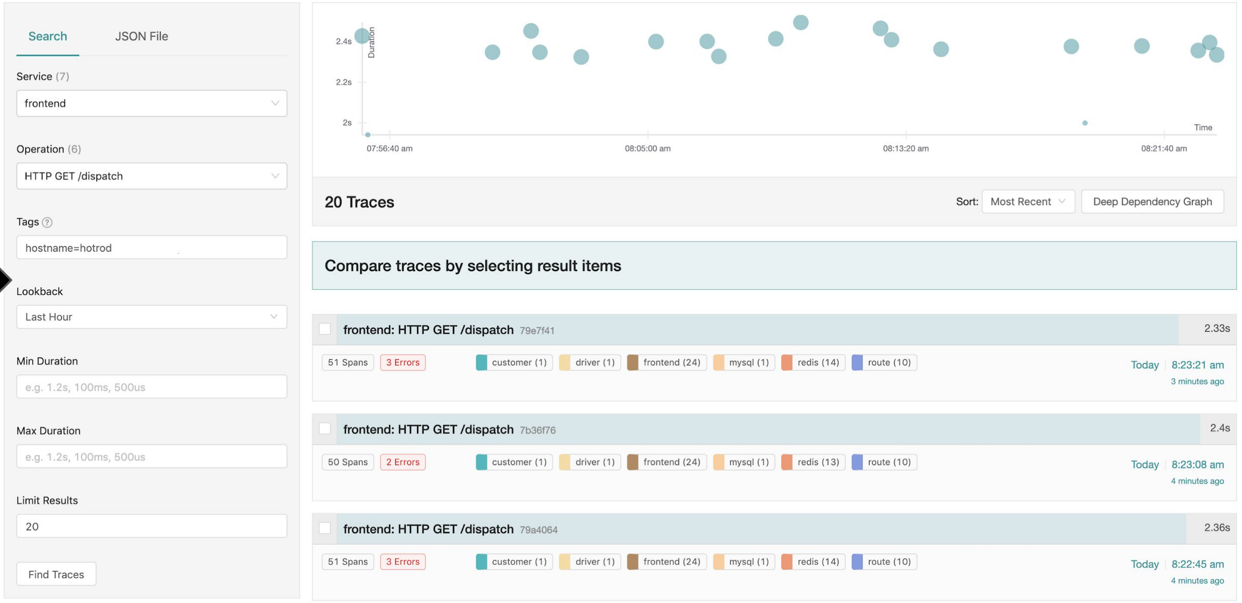
```
Common labels: {app="cassandra", controller_revision_hash="cassandra-sfs-c6dcffd97",
job="metrictank/cassandra", namespace="metrictank"}
Limit: 1000 (228 returned)

2018-11-09 18:30:23  {app="cassandra", controller_rev…  INFO [RMI TCP Connection(2556)-127.0.0.1] 2018-11-09 18:30:23,927
                                                        NoSpamLogger.java:91 - Maximum memory usage reached (512.000MiB), cannot
                                                        allocate chunk of 1.000MiB
2018-11-09 18:30:16  {app="cassandra", controller_rev…  WARN [PERIODIC-COMMIT-LOG-SYNCER] 2018-11-09 18:30:16,913
                                                        NoSpamLogger.java:94 - Out of 33 commit log syncs over the past 278.18s
                                                        with average duration of 39.42ms, 2 have exceeded the configured commit
                                                        interval by an average of 18.05ms
18-11-09 18:27:57  {app="cassandra", controller_rev…  INFO [RMI TCP Connection(2060)-127.0.0.1] 2018-11-09 18:27:57,142
                                                        NoSpamLogger.java:91 - Maximum memory usage reached (512.000MiB), cannot
                                                        allocate chunk of 1.000MiB
-09 18:25:16  {app="cassandra", controller_rev…  WARN [PERIODIC-COMMIT-LOG-SYNCER] 2018-11-09 18:25:16,846
                                                        NoSpamLogger.java:94 - Out of 35 commit log syncs over the past 291.78s
                                                        with average duration of 52.94ms, 4 have exceeded the configured commit
                                                        interval by an average of 22.43ms
-09 18:25:16  {app="cassandra", controller_rev…  WARN [PERIODIC-COMMIT-LOG-SYNCER] 2018-11-09 18:25:16,813
```

Integrations - Taking a closer look

If you take a closer look at how jumping between metrics and traces is today, they are generally linked by common set of labels and time window.





Service (7): frontend

Operation (6): HTTP GET /dispatch

Tags: hostname=hotrod

Lookback: Last Hour

Min Duration: e.g. 1.2s, 100ms, 500us

Max Duration: e.g. 1.2s, 100ms, 500us

Limit Results: 20

20 Traces

Compare traces by selecting result items

frontend: HTTP GET /dispatch 79e7f41 — 2.33s
51 Spans | 3 Errors | customer (1) | driver (1) | frontend (24) | mysql (1) | redis (14) | route (10) | Today 8:23:21 am, 3 minutes ago

frontend: HTTP GET /dispatch 7b36f76 — 2.4s
50 Spans | 2 Errors | customer (1) | driver (1) | frontend (24) | mysql (1) | redis (13) | route (10) | Today 8:23:08 am, 4 minutes ago

frontend: HTTP GET /dispatch 79a4064 — 2.36s
51 Spans | 3 Errors | customer (1) | driver (1) | frontend (24) | mysql (1) | redis (14) | route (10) | Today 8:22:45 am, 4 minutes ago

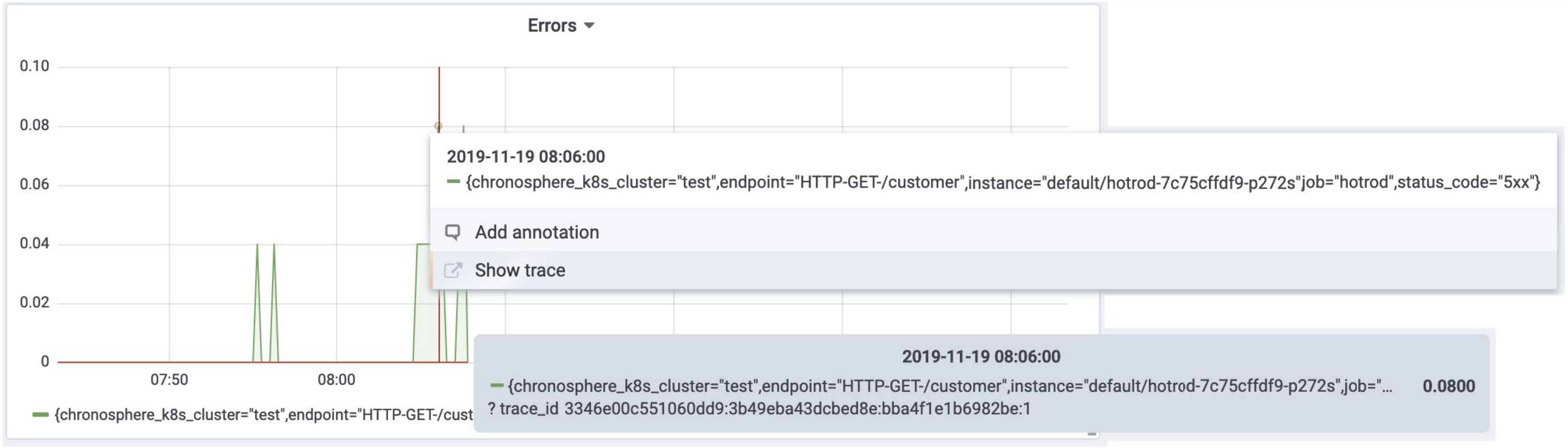
Integrations - Taking a closer look

This narrows down search space in terms of time window and labels, but:

- Querying metrics with sum(...) or any other aggregation will drop tags
    - Context lost for jumping from metrics to traces
- Only “magical” when you store every trace
    - For a lot of users is prohibitively expensive.
- When sampling, the chances of having the right trace is low
    - Especially debugging edge cases - P99, or one error in a thousand.

Wouldn’t it be nice if?

Go straight from the metric datapoint to one of the traces for a request that comprised that **exact datapoint**.



2019-11-19 08:06:00
{chronosphere_k8s_cluster="test",endpoint="HTTP-GET-/customer",instance="default/hotrod-7c75cffdf9-p272s"job="hotrod",status_code="5xx"}

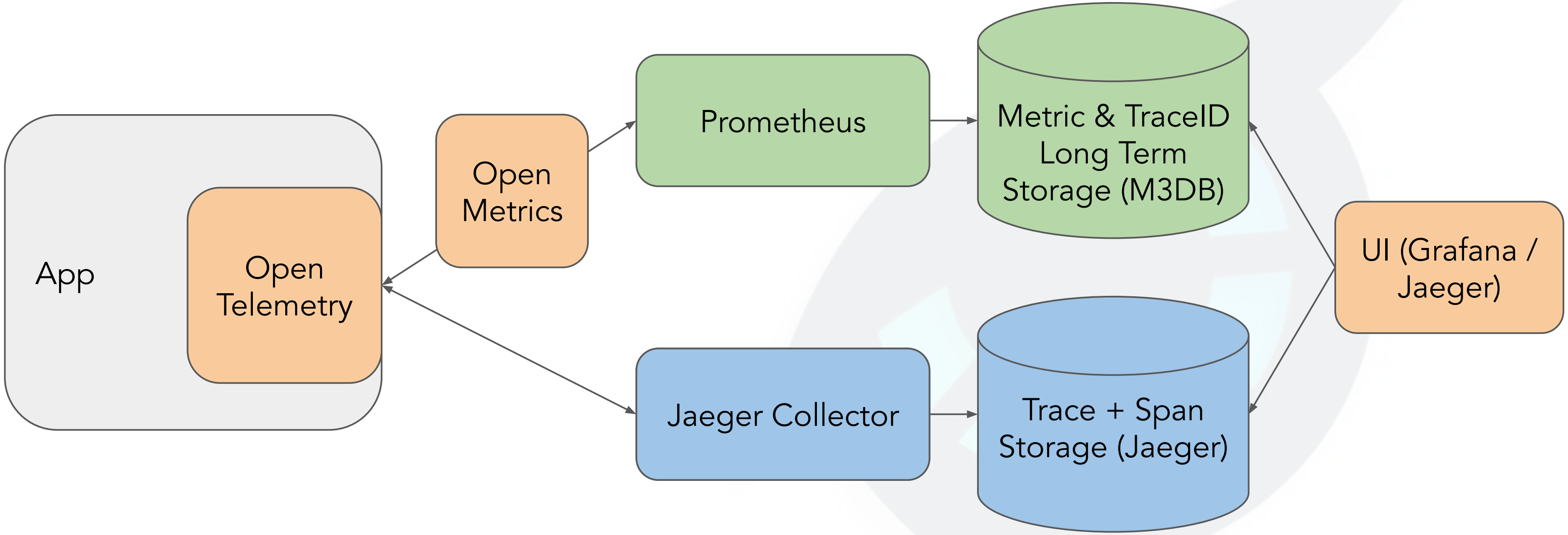
2019-11-19 08:06:00
{chronosphere_k8s_cluster="test",endpoint="HTTP-GET-/customer",instance="default/hotrod-7c75cffdf9-p272s",job="... 0.0800
? trace_id 3346e00c551060dd9:3b49eba43dcbed8e:bba4f1e1b6982be:1

{chronosphere_k8s_cluster="test",endpoint="HTTP-GET-/cust

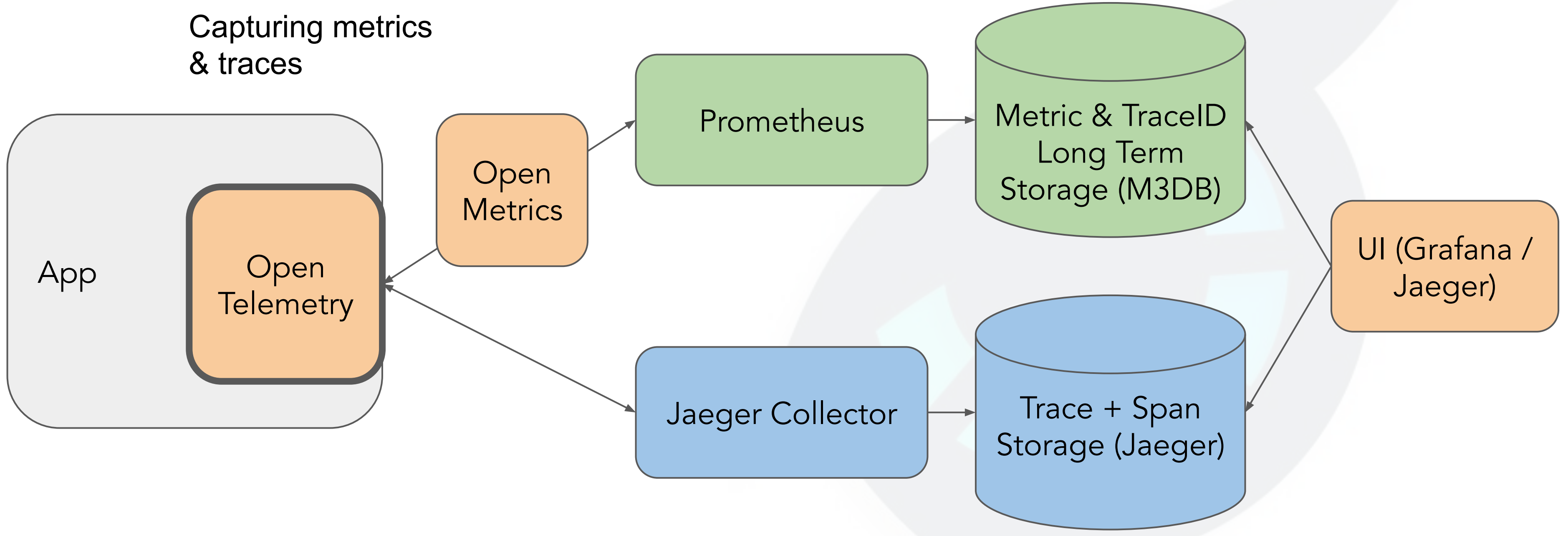
3 Deep Linking Metrics and Traces

Demo

Putting it altogether



What is OpenTelemetry?



Capturing metrics & traces
App
Open Telemetry
Open Metrics
Prometheus
Metric & TraceID Long Term Storage (M3DB)
Jaeger Collector
Trace + Span Storage (Jaeger)
UI (Grafana / Jaeger)

OpenTelemetry: Instrumentation SDK

```go
jobsQueuedGauge := meter.NewFloat64Gauge("jobs_queued",
    metric.WithDescription("The number of jobs currently queued"))


err := tracer.WithSpan(ctx, "jobEnqueue", func(ctx context.Context) error {
    jobsTotal, err := jobQueue.Enqueue(job)
    if err != nil {
        return err
    }
    jobsQueuedGauge.Set(ctx, jobsTotal)
})
```

CLOUD NATIVE
COMPUTING FOUNDATION

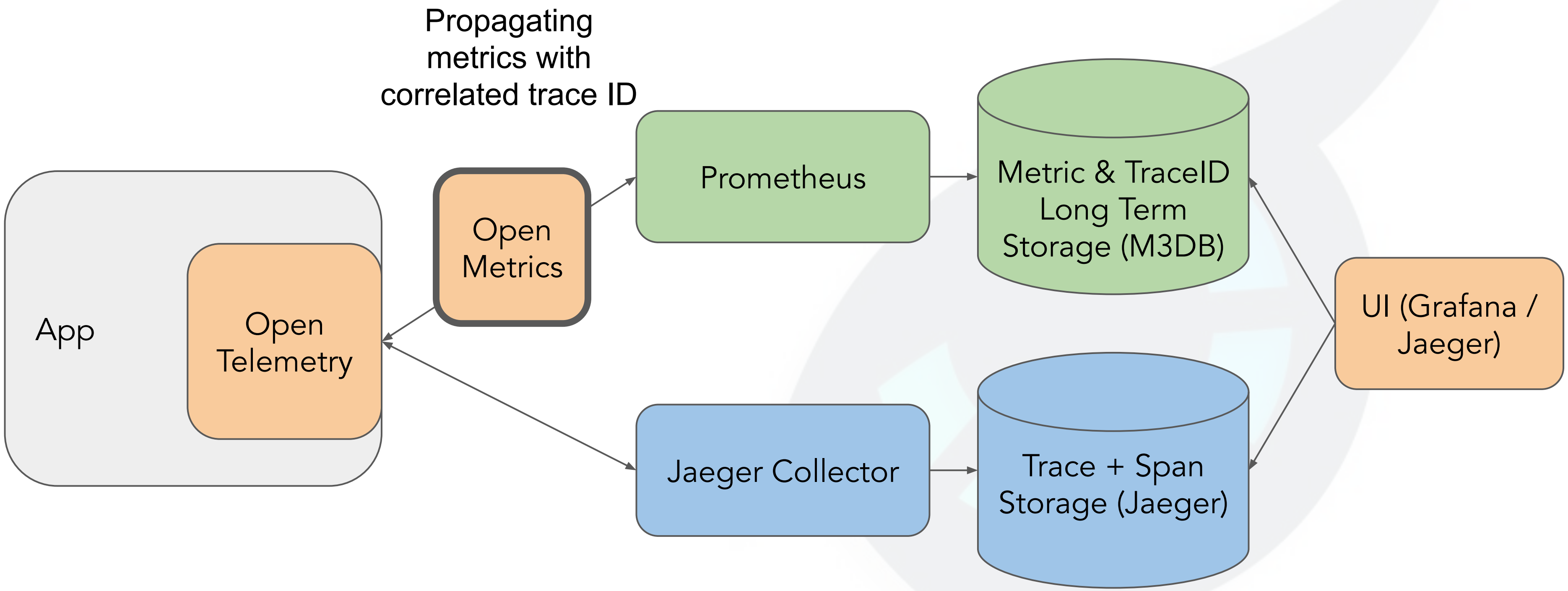
OpenTelemetry: Instrumentation SDK

```go
jobsQueuedGauge := meter.NewFloat64Gauge("jobs_queued",
    metric.WithDescription("The number of jobs currently queued"))


err := tracer.WithSpan(ctx, "jobEnqueue", func(ctx context.Context) error {
    jobsTotal, err := jobQueue.Enqueue(job)
    if err != nil {
        return err
    }
    jobsQueuedGauge.Set(ctx, jobsTotal)
})
```

CLOUD NATIVE
COMPUTING FOUNDATION

What is OpenMetrics?

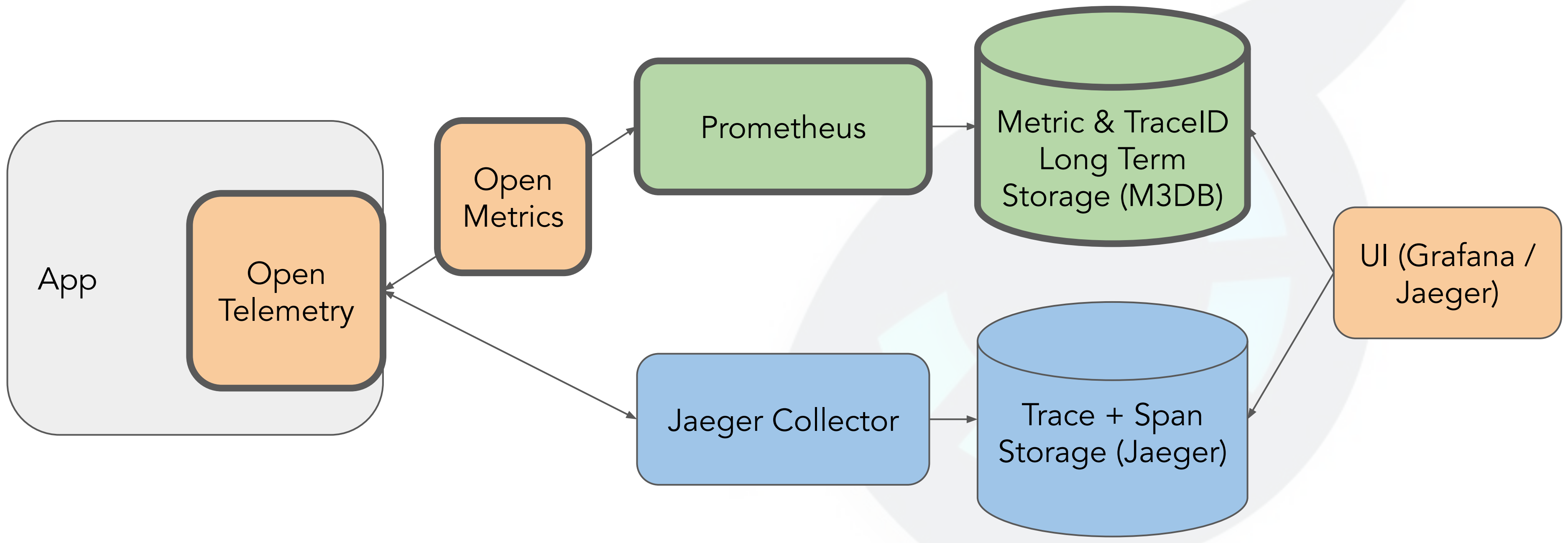

Propagating metrics with correlated trace ID
App
Open Telemetry
Open Metrics
Prometheus
Metric & TraceID Long Term Storage (M3DB)
Jaeger Collector
Trace + Span Storage (Jaeger)
UI (Grafana / Jaeger)

OpenMetrics: Extended Prometheus exposition

```
# HELP http_requests_total http_requests
# TYPE http_requests_total counter
http_requests_total{endpoint="/search",status_code="2xx"} 1725 # {trace_id="b096e71d..."} 1
http_requests_total{endpoint="/search",status_code="4xx"} 4 # {trace_id="944a6d97..."} 1
http_requests_total{endpoint="/search",status_code="5xx"} 27 # {trace_id="50785260..."} 1
http_request_latency_bucket{endpoint="/search",le="0.1"} 7 # {trace_id="7f78deda..."} 1
http_request_latency_bucket{endpoint="/search",le="0.2"} 7 # {trace_id="5ad53ac9..."} 1
http_request_latency_bucket{endpoint="/search",le="0.3"} 7 # {trace_id="c78493ec..."} 1
...
```

CLOUD NATIVE
COMPUTING FOUNDATION

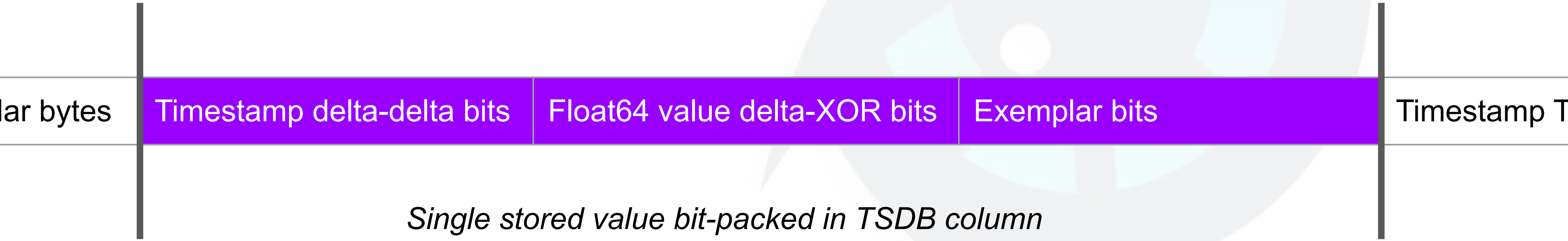
Putting it all together



Prometheus and M3

Prometheus scrapes the exemplar, keeps it locally in memory, then remote writes it to M3.

M3DB stores the trace ID next to the metric timestamp and float value.



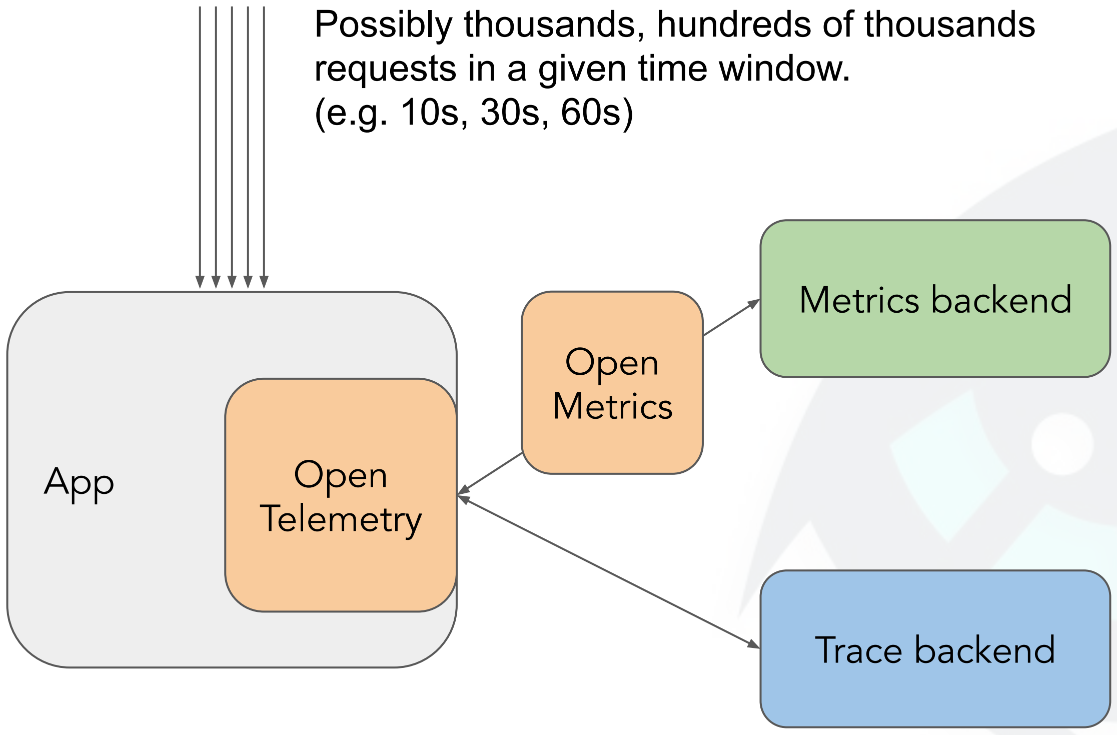
Single stored value bit-packed in TSDB column

How do we query it?

When querying the data M3 query makes sure to keep at least one representative exemplar per datapoint as part of the result (even after applying sum(...), histogram_quantile(...), etc)

```
{
  "metric": {
    "chronosphere_k8s_cluster": "test",
    "endpoint": "HTTP-GET-/customer",
    "instance": "default/hotrod-7c75cffdf9-p272s",
    "job": "hotrod",
    "status_code": "5xx"
  },
  "values": [
    [
      1574178960,
      "5",
      "trace_id:ac51ab1117abfdc:d4c6a84f5833856:3638a233f5a36b71:1"
```

Efficiency/Scalability?



Possibly thousands, hundreds of thousands requests in a given time window. (e.g. 10s, 30s, 60s)

Single datapoint stored with a single trace ID for error counter value or latency histogram bucket representing thousands of requests. (efficient)

Need to store all thousands of traces to guarantee the trace ID picked for datapoint exists. (very inefficient & expensive)

Wouldn’t it be nice if?

200 Status Code	400 Status Code	500 Status Code
{trace: 024253eb-6be0-...}	{trace: a0eb52dc-8a3e-...}	{trace: a22476ff-b177-...}
{trace: 841e6da2-8694-...}	{trace: f86aa034-b7c5-...}	
{trace: f7e33019-abc8-...}	{trace: 6aa9d08f-6632-...}	
{trace: 7b2a9954-e213-...}	{trace: 1be7ef05-9985-...}	
{trace: d37ce450-a463-...}		
{trace: b78fe85b-a508-...}		
{trace: b10964a4-a4af-...}		
{trace: 1cdfcb7a-1849-...}		
{trace: 3bb247b2-89ec-...}		
{trace: 2bce5524-905e-...}		

Guaranteeing one representative trace stored?

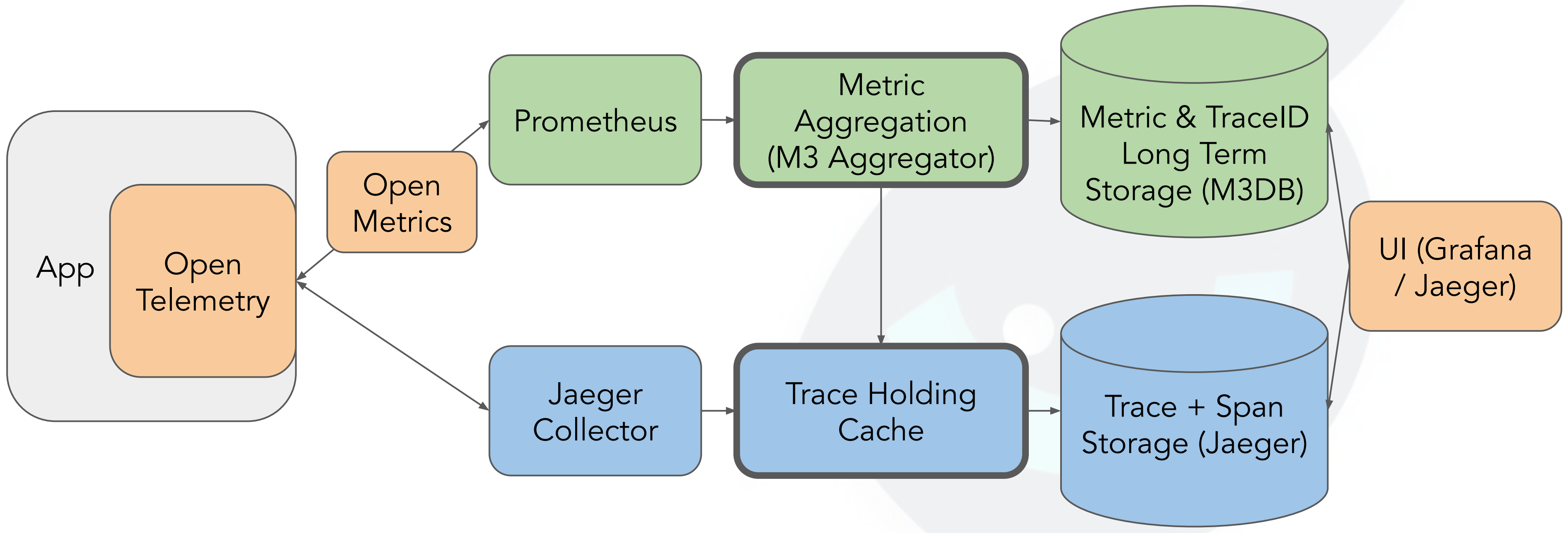
Using metric aggregation to determine sampled traces has many upsides:

- Aggregate across time to collect traces at **useful time intervals**.
- Metric tags are great at capturing all **unique combinations**. eg:

  Error/Success, ErrorStatusCode, LatencyBucket.

  - This also ensures unique combinations of traces.
- Allows maintaining **direct link** between metric datapoint and trace ID.

Putting it altogether (again)


App
Open Telemetry
Open Metrics
Prometheus
Metric Aggregation (M3 Aggregator)
Metric & TraceID Long Term Storage (M3DB)
Jaeger Collector
Trace Holding Cache
Trace + Span Storage (Jaeger)
UI (Grafana / Jaeger)

Where can I get this? (hint: upstream in progress)

Current end-to-end demo at:

https://github.com/chronosphereio/demo-deeplink-metrics-traces

Merged: Add exemplar support to OpenMetrics:

https://github.com/prometheus/prometheus/pull/6292

Open(needs discussion): Store exemplars in Prometheus memory, forward on remote write:

https://github.com/prometheus/prometheus/pull/6309

Open(helping review): OpenMetrics/Prometheus exporter PR for OpenTelemetry:

https://github.com/open-telemetry/opentelemetry-go/pull/334

Where can I get this?

OpenMetrics https://github.com/OpenObservability/OpenMetrics

OpenTelemetry https://github.com/open-telemetry/opentelemetry-specification

Prometheus https://github.com/prometheus/prometheus

M3 https://github.com/m3db/m3

Grafana https://github.com/grafana/grafana

Talk demo https://github.com/chronosphereio/demo-deeplink-metrics-traces

Thank you and Q&A

Come say hi! Booth SE62.

chronosphere

@roskilli

http://bit.ly/m3slack